Manual PICAXE

www.picaxe.co.uk

Tradução livre e parcial (adaptada ao PICAXE 28X) por Ludgero Leote

revolution

O que é um microcontrolador?

O microcontrolador é por vezes descrito como "computador num *chip*".

Trata-se de um circuito integrado de baixo custo que contem memória, unidade de processamento e circuitos de entradas/saídas num mesmo circuito integrado.

Os microcontroladores são adquiridos "limpos" e programados pelo utilizador com software específico para uma dada tarefa.

Uma vez programado, o microcontrolador é inserido num produto para o tornar mais inteligente e fácil de usar.

Tome-se, como exemplo, o forno de micro-ondas, onde um microcontrolador trata a informação proveniente do teclado, mostra as informações no *display* e controla os dispositivos de saída (motor do prato-rotativo, luz, avisador sonoro, magnetrão). Um microcontrolador pode frequentemente substituir uma quantidade de componenets separados, ou memso um circuito electrónico completo.

Algumas das vantagens de utilizar microcontroladores no *design* de um produto são:

- elevada fiabilidade
- níveis de armazenamento reduzidos, pois um microcontrolador substitui vários componentes
- montagem simplificada do produto e redução do tempo de fabrico
- maior flexibilidade de produtos e adaptabilidade pois as características do produto são programadas no microcontrolador e não embutidas no *hardware* electrónico.
- modificações rápidas no produto e seu desenvolvimento por alteração do programa e não do *hardware* electrónico.

Algumas aplicações dos microcontroladores são na aparelhagem doméstica, nos sistemas de alarme, nos equipamentos médicos, nos subsistemas dos veículos automóveis, instrumentação electrónica, telecomunicações, etc.

Alguns dos modernos automóveis utilizam mais de trinta microcontroladores – em subsistemas como sistema de injecção, ar condicionado, alarme, sinalização, air-bags, ABS, etc.

Na indústria os microcontroladores são usualmente programados em linguagens assembly ou C.

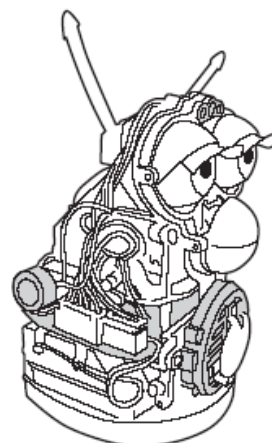
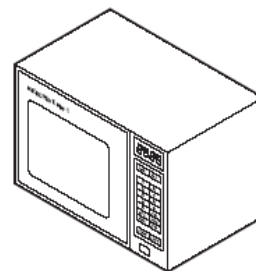
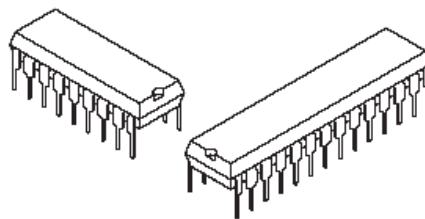
Contudo, face à complexidade destas linguagens, não é realista o seu uso com jovens estudantes no ensino, ou por curiosos sem treino formal.
O sistema PICAXE ultrapassa este problema pelo uso de uma linguagem com uma curva de aprendizagem mais rápida, a linguagem BASIC. Os programas podem ainda ser especificados graficamente utilizando um editor de fluxogramas.

Microcontroladores, input e outputs

Na imagem ao lado é mostrado um popular brinquedo de há alguns anos, o Furby. Trata-se de um excelente exemplo de um sistema mecatrónico, pois utiliza um circuito de controlo electrónico para controlar um elevado número de mecanismos. Contem ainda um elevado número de sensores pelo que pode reagir a mudanças quando é movido (por exemplo, quendo é colocado num local escuro ou virado de cabeça-para-baixo).

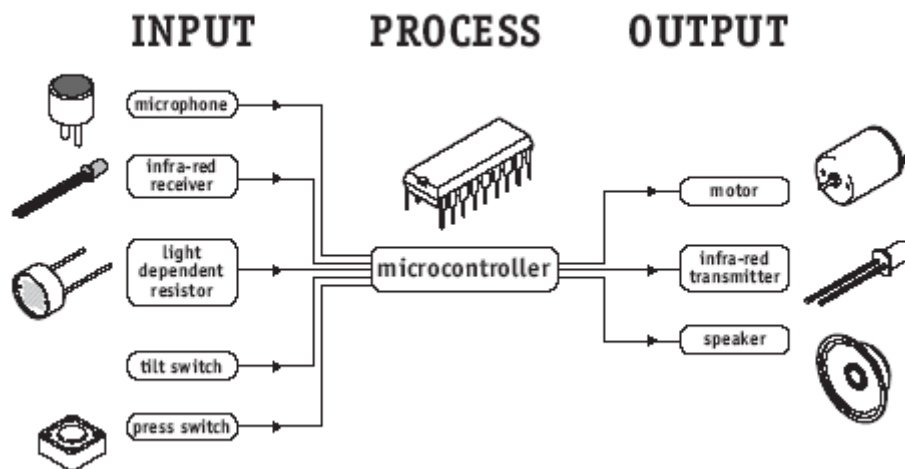
Os transdutores de Input (entrada) são dispositivos electrónicos que detectam alterações no "mundo real" e enviam sinais para o bloco de processamento do sistema electrónico.

Alguns dos transdutores de entrada deste brinquedo são:

- interruptores de pressão à frente e a trás para detectar se o brinquedo é "acariciado" e um interruptor na boca para detectar quando é "alimentado".

- uma resistência dependente da luz (LDR) entre os olhos para detectar é dia ou noite,
- um microfone para detectar ruídos e falas.
- um interruptor de inclinação para detectar quando o brinquedo é deitado ou virado.
- Um detector de infra-vermelhos para detectar sinais enviados por outros brinquedos.

Os transdutores de Output (saída) são dispositivos electrónicos que podem ser accionados (ligados) pelo bloco de processamento do sistema electrónico. Alguns dos transdutores de saída neste brinquedo são:

- um motor para fazer mover os olhos e a boca.
- um altifalante para produzir sons.
- um LED de infra-vermelhos para enviar sinais para outros brinquedos.

O microcontrolador utiliza a informação dos transdutores de entrada para tomar decisões sobre como controlar os dispositivos de saída. Estas decisões são tomadas pelo programa de controlo, que é transferido (downloaded) para o microcontrolador. Para modificar o "comportamento" do brinquedo basta proceder a alterações no programa e voltar a enviá-lo para o microcontrolador.

O que é um sistema PICAXE?

O sistema PICAXE explora as característica singulares da nova geração de microcontroladores de baixo custo com memória FLASH. Estes microcontroladores pode ser programados uma vez e outra (tipicamente 100000 vezes) sem a necessidade de programadores caros.

O PICAXE utiliza uma linguagem BASIC simples (ou fluxogramas gráficos) que podem ser usados por jovens estudantes para se iniciarem na criação de programas uma hora depois de começarem. É muito mais fácil aprender e detectar erros do que com linguagens de programação como o C ou o assembly.

Ao contrário de outros sistemas baseados em "módulos" BASIC, toda a programação do PICAXE é realizada ao nível do "chip". Assim, ao contrário de comprar um módulo caro pré-assemblado (de difícil reparação), com um sistema PICAXE pode simplesmente adquirir um chip standard e usá-lo directamente na placa do seu projecto.

A potência do sistema PICAXE reside na sua simplicidade. Não é necessário programador, apagador ou sistemas electrónicos complicados – o microcontrolador é programado através de um cabo série com três condutores ligado a um PC. Um sistema funcional PICAXE é constituído por 3 componentes e pode ser construído num *breadboard*, *stripborad* ou placa de circuito impresso.

O software PICAXE 'Programming Editor' é gratuito pelo que o único custo reside no cabo de *download*. Num ambiente educativo isto facilita a possibilidade de os estudantes comprarem o seu próprio material e às escolas equiparem cada computador com um cabo. Outros sistemas que exigem programadores ou "módulos" caros são normalmente excessivamente caros de implementar.

Finalmente, como o *chip* PICAXE nunca é retirado da placa de projecto, não existem danos nos pinos (que ocorrem facilmente cada vez que se retira um microcontrolador da placa para o programador).

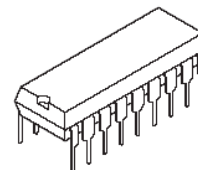
Construindo o próprio circuito / PCI

O sistema PICAXE foi desenhado para permitir que estudantes/curiosos construam os seus próprios circuitos. Contudo, se não quiser construir o seu próprio circuito, existe uma enorme variedade de placas em PCI disponíveis - ver catálogo on-line para mais detalhes.

Se quiser fazer o seu próprio circuito, siga as indicações disponíveis no sítio na Internet www.picaxe.co.uk.

O que é um microcontrolador PICAXE?

Um microntrolador PICAXE é um microcontrolador *standard* da Microchip PICmicro™ que foi previamente pré-programado com código *bootstrap*.

O código *bootstrap* possibilita que o microcontrolador possa ser programado através de uma ligação série ao PC. Isto elimina a necessidade de um programador convencional (e caro), tornando todo o sistema muito barato.

O código *bootstrap* pré-programado contem ainda rotinas comuns (como a que gera atrasos ou saídas para som), pelo que cada transferência de programa não necessita de perder tempo a carregar esse código. Isso torna a transferência de programas para o microcontrolador muito rápida.

Como os microcontroladores não-programados comprados para *fazer* microcontroladores PICAXE são adquiridos em grandes quantidades, é possível aos fabricantes vender os PICAXE a preços próximos do microcontrolador não-programado. O código *bootstrap* PICAXE não é disponibilizado para programação de microcontroladores virgens.

Utilizando o sistema PICAXE

Para usar o sistema PICAXE é necessário possuir:

• Um microcontrolador PICAXE
• Uma placa de circuito impresso PICAXE ou uma *breadborad/stripboard*.
• Uma fonte de alimentação (i.e.. 4 baterias recarregáveis AA (4.8V) ou 3 pilhas alcalinas AA (4.5V))
• Um cabo série para *download*.
• O software gratuito 'Programming Editor'.

Todos estes items estão contidos nos *packs* para iniciação da PICAXE.

Para correr o software é necessário possuir um computador com sistema operativo Windows 95 ou posterior. Qualquer sistema que possua o S.O. Windows funcionará no modo BASIC, embora para programação no modo gráfico (fluxogramas) seja aconselhado um Pentium 4 ou superior.

O computador deve possuir um porto de comunicação série de 9 pinos para ligar o cabo de transferência de dados. No caso dos novos portáteis que não possuem porto série, é necessário adquirir um cabo de conversão USB/série. Veja também a a secção referente à definição do porto série para mais informações.

Para comparação entre as características dos diferentes chips PICAXE e para informação sobre as várias placas C.I. disponíveis para iniciação, veja, por favor, o original da Revolution em picaxe_manual1.pdf.

Instalação do Software

Características do computador

Para instalar o software é necessário um computador com sistema operativo Windows 95 ou superior com aproximadamente 20MB de espaço livre em disco. Qualquer computador que corra o sistema operativo Windows funcionará no modo 'BASIC'. Contudo para programação no modo fluxograma, é necessário no mínimo um Pentium 4.

Instalação:

1) Ligue o computador e autentique-se (alguns sistemas operativos exigem autorização do administrador para instalção do software – contacte o administrador do sistema).
2) Insira o CD, ou faça *download* do sítio Internet da PICAXE em www.picaxe.co.uk e execute o ficheiro de instalação do software.
3) Siga as instruções do ecrã para instalar o software. Nos computadores mais antigos pode ser necessário reiniciar o computador para completar a instalação.
4) Insira o cabo série no conector série de 9 pinos do computador. Se o computador for portátil e recente poderá não possuir esse conector. Nesse caso terá que instalar previamente o cabo de conversão USB/série com o software que o acompanha (veja a secção Instalação do conversor USB/série). Anote qual o número do porto série utilizado 'COM' (geralmente COM1 ou COM2).
5) Clique em **Start>Programs>Revolution Education>Programming Editor** para executar o software. Se o ecrã Options não aparecer automaticamente, clique no menu **View>Options**. No separador 'Mode' seleccione o tipo e versão de microcontrolador (neste caso PICAXE 28X a 4MHz). No separador 'Port' seleccione o porto série COM apropriado e prima OK.

Está pronto para usar o PICAXE.

Instalação do conversor USB/série

A maior parte dos computadores *desktop* possuem um conector série de 9 pinos para ligação do cabo de transferência de ficheiros (*download*) do PICAXE. Contudo, a maior parte dos actuais computadores portáteis não possuem esse conector, mas sim conectores USB. O sistema de interface USB é um sistema inteligente que o dispositivo ligado se configure quando ligado ao porto USB. Embora seja teoricamente possível fabricar uma versão USB do PICAXE, a memória extra necessária iria encarecer o custo do integrado em cerca de £3 ($5). Assim, utiliza-se um sistema alternativo. O utilizador deverá adquirir um cabo conversor USB/série. Este cabo custa aproximadamente £15 ($20) e pode ser usado para quaisquer outros dispositivos.

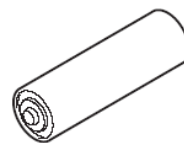
Fonte de alimentação

Todos os chips PICAXE podem funcionar com tensões de alimentação entre 3 e 5.5V CC. Contudo, alguns computadores podem exigir uma alimentação do PICAXE entre 4.5V e 5.5V para que as comunicações se façam correctamente na transferência de ficheiros (*download*). Recomenda-se assim que a fonte de alimentação seja uma das seguintes:

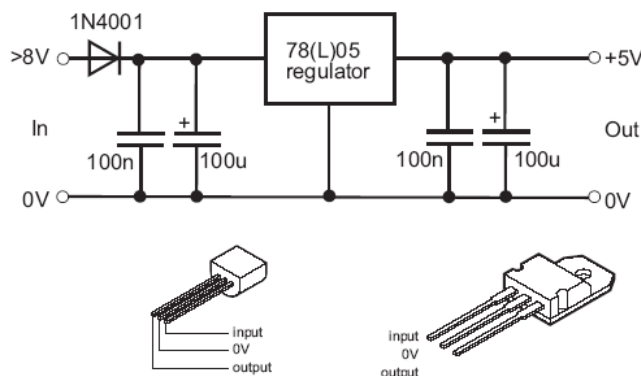
- 3 x AA pilhas alcalinas AA (4.5V)
- 4 x baterias recarregáveis AA (NiCad ou NiMh) (4.8V)
- fonte de alimentação regulada de 5V a partir de 9V CC.

Não deve utilizar baterias ou pilhas de 9V PP3, pois estão muito acima do máximo admitido e podem provocar danos permanentes no PICAXE. As baterias PP3 9V são projectadas para aplicações de baixo consumo de corrente e longa duração (por ex. alarmes ou multímetros). Embora uma bateria PP3 9V regulada para 5V possa funcionar por curtos períodos na alimentação do microcontrolador, assim que forem ligados dispositivos às saídas (por ex. leds, motores, bezouros, etc.) irá rapidamente descarregar. Deverá portanto utilizar *packs* de baterias e não pilhas 9V PP3 em projectos de microcontroladores.
Tenha cuidado na inserção dos integrados PICAXE nos circuitos assegurando-se de que estão na posição correcta, pois a inversão dos pinos pode provocar danos permanentes.

Packs de bateria AA

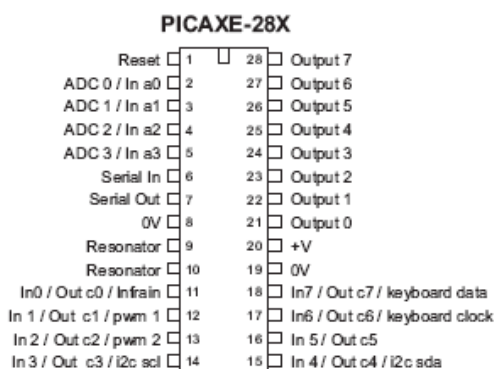
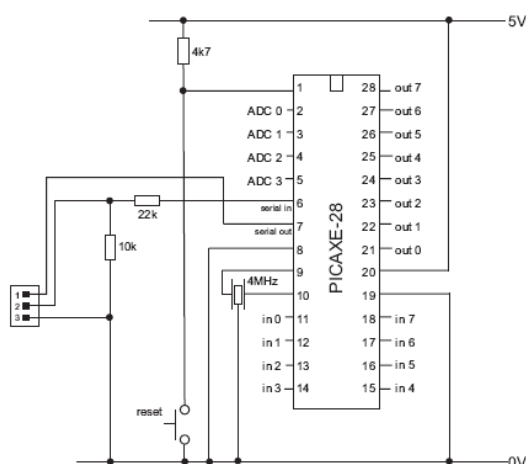
As pilhas Alcalinas AA possuem uma tensão nominal de 1.5V, pelo que bastam 3 unidades para obter os 4.5V mínimos da alimentação. Se utilizar 4 unidades (4 x 1,5V=6V) já terá que inserir um díodo 1N4001 em série para reduzir a tensão. O díodo além de proteger de inversão de polaridade provoca uma queda de tensão de 0,7V, logo obterá uns aceitáveis 5.3V (6V-0.7V).
As baterias recarregáveis AA (NiCad e NiMh) possuem uma tensão nominal de 1.2V, pelo que 4 unidades produzem 4.8V.
Tenha cuidado em não curto-circuitar os terminais dos *packs* de baterias, pois a enorme corrente de curto-circuito pode danificá-los produzindo aquecimento ou mesmo o início de um incêndio.

Fontes de alimentação reguladas

Alguns utilizadores podem desejar utilizar fontes de alimentação fixas.É essencialque se utilize uma fonte de qualidade de 9V CC com um regulador de tensão de 5V. As fontes não-reguladas (com cargas reduzidas) produzem tensões excessivas e podem danificar o microcontrolador.

A fonte de alimentação 9V CC deve ser regulada para 5V utilizando um regulador de tensão como o 7805 (1A corrente) ou 78L05 (100mA corrente). O circuito completo do regulador é o apresentado na figura junta.

O díodo1N4001 garante protecção contra a inversão de polaridade e os condensadores ajudam a estabilizar a tensão de 5V. Note que estes reguladores de tensão não funcionam adequadamente senão quando a tensão de entrada é 8V ou superior.

Nota do tradutor: Caso queira pode usar o LM2940 que apenas precisa de 6V para fornecer uma tensão regulada de 5V. Isto é, possibilita o uso de packs 6x1,2V=7,2V sem problemas.

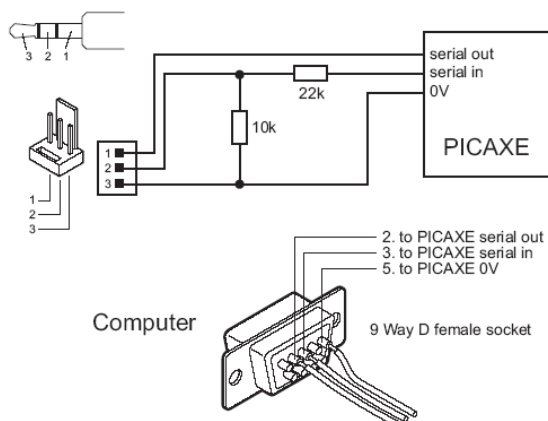
PICAXE-28X Pinout and Circuit

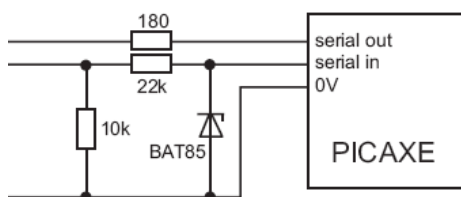
O diagrama de pinos do PICAXE28X é o seguinte:

O circuito electrónico mínimo para utilizar o PICAXE28X é:

Veja a secção Circuito série para transferência de dados (*download*) para mais detalhes.

Notas:
1) As resistências 10k/22k são essenciais.
2) O pino de *reset* pode ser ligado directamente através de uma resistência de 4k7 ohms aos 5V.
3) É necessário um ressoador cerâmico de 4MHz.

Circuito série de transferência de dados (*download*)

O circuito série para transferência de dados (download) é idêntico para todos os chips PICAXE. É constituído por 3 condutores que vão do PICAXE para o porto série do computador. Um dos condutores transmite dados do computador para o microcontrolador, outro transmite dados da saída de dados do microcontrolador para o computador e o terceiro é a massa comum (referência).

O circuito mínimo é apresentado na figura junta. Este circuito é adequado para a maior parte das aplicações educativas.

Note que as duas resistências constituem um divisor de tensão. A resistência de 22k, juntamente com os díodos internos do microcontrolador, adaptam a tensão de saída série à alimentação do PICAXE, limitando a corrente a valores aceitáveis. A resistência de 10k bloqueia a flutuação da entrada série enquanto o cabo série não é ligado. As duas resistências devem ser incluídas em qualquer projecto com circuitos PICAXE (não estão incluídas no cabo série).
A entrada série não deve ficar desligada. Caso fique desligada a entrada série irá *flutuar* entre alto e baixo, provocando mau funcionamento – o PICAXE vai interpretar essa flutuação como transferência de dados.

Circuito série melhorado para transferência de dados

O díodo Shottky BAT85 funciona com tensão mais reduzida do que os díodos internos do microcontrolador, estabelecendo uma tensão de referência mais precisa. A resistência adicional de180R garante uma protecção adicional contra curto-circuitos no pino de saída série.

Cabos para transferência de dados (*download*)

O cabo série de transferência de dados é constituído por um cabo de 4 condutores (TVHV ou equivalente), possui uma ficha standard 3 pinos Molex 0.1" (2.54 mm) num dos terminais e uma ficha série D 9 pinos macho na outra extremidade. São apenas ligados os pinos 2, 3 e 5 da ficha D.

Esquema de ligações

Ficha Molex **Pino do PICAXE**	**Ficha D série**
Saída série (7)	pino 2
Entrada série (6)	pino 3
Massa (8)	pino 5

No caso de o computador não possuir porto série deve usar um adaptador USB/série. Para computadores mais antigos com ficha série de 25 pinos, vai precisar de um adaptador 25-9 pinos.

Circuito de *Reset*

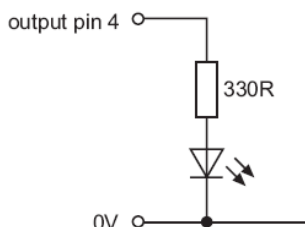
Todos os PICAXE de 18, 28 e 40 pinos possuem um pino de ‘reset’. Este pino deve estar no estado alto para que o microcontrolador funcione. Se o pino ficar desligado o microcontrolador não funciona. Para ligá-lo basta inserir uma resistência xde 4k7 entre o pino e a alimentação de +5V. Opcionalmente pode incluir um microswitch entre o pino e a massa (0V) – isso permite-lhe reinicializar o micocontrolador.

Ressoador

Todos os PICAXE de 28 e 40 pinos necessitam de um ressoador (ou cristal de quartzo) externo.
Recomenda-se o uso de um ressoador cerâmico 4MHz 3 pinos (referência RES035).
Este dispositivo é constituído por um ressoador e dois condensadores num único invólucro de 3 pinos (N.T - caso opte por um cristal de quartzo, mais caro mas mais preciso, terá que incluir os dois condensadores de 22pF entre os pinos terminais e a massa). O pino central é ligado à massa (0V) e os outros dois pinos, indiferentemente, aos pinos respectivos do PICAXE (9 e 10 no PICAXE28X).

Caso necessário pode fazer *overclock* no PICAXE usando ressoadores de 8MHz ou 16MHz.Veja a secção 'Over-clocking' para mais detalhes.

Testando o sistema

Este primeiro e simples programa pode ser utilizado para testar o sistema. Requere a ligação de um LED (e uma resistência de 330R em série) ao pino 4 (assegure-se da polaridade correcta do LED).

1. Ligue o cabo do PICAXE ao porto série do computador. Registe qual o número do porto a que está ligado (normalmente designado por COM1 ou COM2).
2. Execute o software Programming.
3. Seleccione **View>Options** para visualizar a janela Options (em princípio aparece automaticamente).
4. Prima no separador 'Mode' e seleccione o tipo de PICAXE correcto.
5. Prima no separador 'Serial Port' e seleccione o porto série onde o cabo do PICAXE está ligado.
6. Prima 'OK'
7. Escreva o programa seguinte:

```
main: high 4
pause 1000
low 4
pause 1000
goto main
```

(NOTA: repare no sinal (:) a seguir à *label* 'main' e nos espaços entre as instruções e os números operandos).

8. Verifique se o circuito do PICAXE está ligado ao cabo série, e de que as baterias estão ligadas. Verifique se o LED e a resistência 330R estão ligadas à saída 4.
9. Seleccione **PICAXE>Run**

Deverá aparecer no ecrã uma barra de *download* enquanto o programa é transferido. Quando a transferência terminar o programa deverá começar a executar-se no PICAXE imediatamente – o LED da saída 4 deverá piscar *on* e *off* segundo a segundo.
Se a transferência de dados não tiver lugar verifique a lista de causas possíveis e efectue um *hard-reset* como indicado na secção seguinte

Procedimento *Hard-reset*

O processo de trnasferência de dados (*download*) chama o PICAXE permanentemente testando a linha de entrada série em busca de um novo sinal vindo do computador. Isto processa-se automaticamente e não é notado pelo utilizador. Contudo, em raras ocasiões pode o PICAXE não ler com rapidez suficiente a linha série enquanto executa um programa.
Estas situações verificam-se quando:

- Existe um programa corrompido no PICAXE (remoção da alimentação ou do cabo durante a transferência)
- Instruções *pause* ou *wait* mais longas que 5 segundos usadas no programa.
- Utilização das instruções *serin, infrain* ou *keyin* no programa.

Mesmo assim, é muito simples resolver este problema, pois a primeira coisa que qualquer PICAXE executa quando há um *reset* é verificar se se trata de uma nova transferência de dados (*download*). Portanto, se fizer *reset* ao PICAXE enquanto uma transferência se inicia, a nova transferência é sempre reconhecida. Este processo designa-se por *hard-reset*.

Para realizar um *hard-reset* utilizando o interruptor de *reset.*
1) Prima o interruptor e mantenha-o premido.
2) Seleccione o menu **PICAXE>Run** para se iniciar o *download.*

3) Aguarde até que a barra de progressão apareça no ecrã.
4) Largue o interruptor de *reset*.

Para realizar um *hard reset* utilizando a fonte de alimentação:

1) Desligue a fonte de alimentação.
2) Aguarde até que os condensadores da fonte de alimentação descarreguem (pode demorar até 30 segundos, conforme o circuito).
3) Seleccione o menu **PICAXE>Run** para iniciar a transferência.
4) Aguarde até que a barra de progressão apareça no ecrã.
5) Ligue a fonte de alimentação.

Lista de controlo (*Download CheckList*)

Se não conseguir fazer download, verifique cada um dos items da lista abaixo.
Se o programa falhar a seguir a um download isso deve-se geralmente a falha de alimentação (ou falha de ligação do cabo). Experimente com uma bateria nova.

Microcontrolador PICAXE

• O integrado PICAXE está correctamente inserido no suporte.
• Está a usar um integrado PICAXE (e não um PIC não programado).
• Está a usar um PICAXE avariado (por ex. o *chip* sofreu uma sobretensão ou inversão de polaridade).
• A alimentação provem de uma fonte CC 4.5V a 5.5V DC regulada.
• O pino *reset* está ligado a V+ através de uma resistência 4.7k.
• O ressoador de 3 pinos está correctamente ligado.
• As resistências10k/22k para o circuito série de *download* estão correctamente ligadas.

Software

• Instalou a última versão do software Programming Editor (v4.1.0 ou posterior, veja a página sobre software em www.picaxe.co.uk para informação actualizada)
• O porto série está correctamente seleccionado (menu **View>Options>Port**).
• A velocidade do ressoador está correctamente seleccionada (menu **View>Options>Mode**)
• Não existe software em execução no computador em conflito com o porto série utilizado.

Cabo de transferência (*Download*)

• O cabo está correctamente ligado.
• O suporte está correctamente ligado às resistências 10k/22k.
• Os pinos do suporte estão correctamente soldadas à placa de circuito impresso.
• O cabo está correctamente inserido na ficha série do computador.
• O cabo está devidamente inserido no suporte da placa.

Adaptador USB

• O adaptador USB/série está correctamente configurado para um porto série.
• O adaptador USB/série utiliza o *driver* correcto (procure no sítio www.picaxe.co.uk o *driver* para o efeito).

Compreendendo a memória do PICAXE

A memória do PICAXE é constituída por três áreas diferentes. A quantidade de memória varia conform o tipo de PICAXE.

Memória de Programa

A memória de programa é onde o programa é guardado após uma transferência (*download*). Trata-se de uma memória rápida tipo FLASH, que se pode reprogramar até cerca de 100 000 vezes. O programa não se perde quando se desliga a alimentação, pelo que é executado assim que esta é ligada de novo. Não é normalmente necessário apagar um programa, pois cada novo *download* reprograma toda a memória. No entanto se quiser parar um programa pode utilizar o menu **PICAXE>Clear Hardware Memory** para efectuar o download de um programa "vazio" para o PICAXE.
Num PICAXE28X pode carregar cerca de 600 linhas de programa. Este valor é aproximado, pois cada instrução ocupa espaços diferentes em memória.
Para verificar a memória livre basta seleccionar o menu **PICAXE>Check Syntax**.

Memória de Dados

A memória de dados é um espaço adicional de memória do microcontrolador. Os dados também não são perdidos quando se desliga a alimentação. Em cada transferência de dados (download) esta memória é posta a 0, a não ser no caso de ser usada a instrução EEPROM para carregar dados na memória.
Veja mais detalhes nas descrições das isntruções *EEPROM, read e write*.

RAM (Variáveis)

A memória RAM é usada para guardar dados temporários em variáveis durante a execução do programa. Esta memória perde toda a informação quando se desliga a alimentação. Existem três tipos de variáveis – usos gerais, armazenamento e funções especiais.
Para informações sobre as variáveis matemáticas veja a informação contida na descrição da instrução *let* do Manual de Instruções BASIC.

Variáveis de Usos Gerais (GPR – General Purpose Registers)

Existem 14 variáveis tipo *byte* de usos gerais. Estas variáveis *byte* são designadas b0 a b13. As variáveis tipo *byte* (8 bits) podem guardar números inteiros entre 0 e 255. As variáveis tipo *byte* não podem representar números negativos nem fraccionários e, no caso de ser excedido o referido intervalo 0-255 darão '*overflow*' sem aviso (por ex.. 254 + 3 = 1) (2 - 3 = 255).
Para números maiores podem combinar-se duas variáveis *byt*e de modo a criar uma variável *word*, que é capaz de guardar números inteiros entre 0 e 65535.
Estas variáveis *word* são designadas w0 a w6, e são construídas do seguinte modo:

w0 = b1 : b0
w1 = b3 : b2
w2 = b5 : b4
w3 = b7 : b6
w4 = b9 : b8
w5 = b11 : b10
w6 = b13 : b12

Portanto, o *byte* mais significativo de w0 é b1, e o *byte* menos significativo de w0 é b0.
Para além disso os *bytes* b0 e b1 (w0) podem ser divididos em variáveis *bit*.
As variáveis *bit* podem ser utilizadas onde for necessário guardar um único *bit* (0 ou 1) numa variável

b0 = bit7: bit6: bit5: bit4: bit3: bit2: bit1: bit0
b1 = bit15: bit14: bit13: bit12: bit11: bit10: bit9: bit8

Pode utilizar qualquer variável *word*, *byte* ou *bit* numa expressão matemática ou instrução que utilize variáveis. Deve contudo precaver-se par a possibilidade de acidentalmente usar a mesma variável 'byte' ou 'bit' que está a ser usada como parte de uma variável 'word' noutra variável.
Todas as variáveis de uso geral são inicializadas a 0 quando é feito *reset*.

Variáveis para armazenamento

As variáveis de armazenamento são localizações de memória adicicional atribuídas para armazenamento temporário de dados tipo byte. Não podem ser usadas em cálculos matemáticos, mas podem ser usadas para armazenar temporariamente valores byte através das instruções *peek* e *poke*.
O número de localizações de memória disponíveis varia conforme o tipo de PICAXE.
Para o PICAXE28X são

112 do endereço 80 ao 127 ($50 to $7F) e ainda do endereço 192 ao 255 ($C0 to $FF)

Estes endereços variam de acordo com as especificações técnicas do microcontrolador.
Para informações sobre as as instruções *poke* e *peek* veja a informação contida na descrição da instrução *let* do Manual de Instruções BASIC.

Variáveis para Funções Especiais (SFR)

As variáveis disponíveis para funções especiais dependem do tipo de PICAXE.

PICAXE-28A / 28X / 40X SFR

pins = representa o dado lido no porto de entrada
pins = representa o porto de saída na escrita

infra = variável usada com a instrução *infrain*
keyvalue = outra designaçao para a variável *infra*, usada pela instrução *keyin*

Note que *pins* é uma ‘pseudo’ variável que se pode aplicar quer a portos de entradad como de saída.
Quando usado à esquerda de uma expressão de atribuição de pinos aplica-se ao porto de saída.
Por exemplo,

```
let pins = %11000011
```

vai colocar as saídas 7,6,1,0 altas e as restantes baixas.

Quando usado à direita de uma expressão dxe atribuição de pinos aplica-se ao porto de entrada (porto C no PICAXE28X).
Por exemplo,

```
let b1 = pins
```

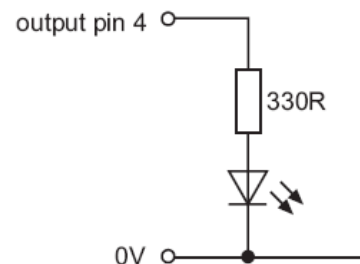
vai guardar em b1 o estado actual do porto de entrada.
Note ainda que

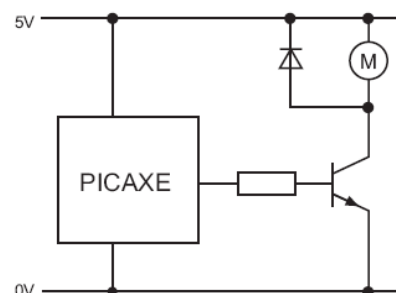
```
let pins = pins
```

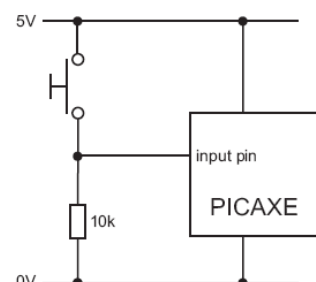
significa ‘faça o porto de saída igual ao porto de entrada’.
A variável pins está separada em variáveis bit individuais para leitura de entradas bit individuais através da instrução if...then.

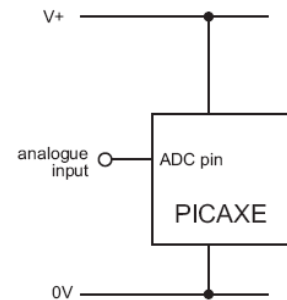
pins = pin7 : pin6 : pin5 : pin4 : pin3 : pin2 : pin1 : pin0

Resumo dos Circuitos para Interface

Esta secção apresenta um breve resumo dos interfaces de entrada/saída com o microcontrolador PICAXE. Para explicações mais detalhadas, veja a secç
3 do Manual de Circuitos de Interface. Nessa secção são fornecidos esquemas de ligação detalhados e programas para a maioria dos dispositivos corrrentes.

Saídas Digitais (Digital Outputs)

O microcontrolador pode deixar passar (sink) ou fornecer (source) correntes de 20 mA nos pinos de saída. Assim dispositivos de baixa baixa corrente, como os LEDs, podem ser directamente ligados aos pinos de saída. Dispositivos que exijam mais corrente podem ser interfaceados através de um transistor (p.ex. BC548B9, um FET ou um Darlington).

Entradas Digitais (Digital Inputs)

Os interruptores podem ser ligados como entradas digitais através de uma simples reistência de 10k. A resistência é essencial pois impede que a entrada fique a “flutuar” quando o interruptor estiver aberto. Isso levaria a um funcionamento imprevisível.

Entradas Analógicas (Analogue Inputs)

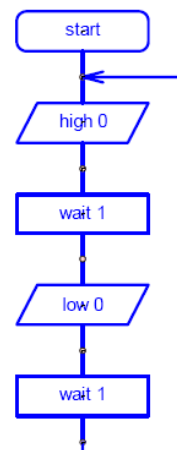
As entradas analógicas podem ser ser ligadas a um divisor potenciométrico entre V+ e 0V. A tensão de referência é a tensão de alimentação, e o sinal analógico não deve exceder a tensão de alimentação.

Fluxogramas ou BASIC?

O software possui dois tipos de programação, a programação por instruções escritas BASIC e a programação gráfica por fluxogramas. Ambos os métodos utilizam as mesmas instruções e sintaxe. O fluxograma constitui um método gráfico simples de juntar as instruções BASIC, prescindindo-se da escrita. Os fluxogramas usam um subconjunto das instruções BASIC e é particularmente destinado ao uso por alunos mais jovens em ambientes escolares.
Uma vantagem da programação por fluxogramas é o ambiente gráfico de simulação. Isso permite que os alunos "vejam" o seu programa ser executado antes de o transferirem para o microcontrolador. Contudo, apenas algumas instruções são suportadas pelo editor.

As maioria dos iniciados e utilizadores educativos preferem as instruções BASIC como método de programação. Trata-se de um sistema mais potente do que o método dos fluxogramas, que se tornam complexos para programas de certa dimensão.

Os fluxogramas são automaticamente convertidos para instruções BASIC antes do programa ser transferido para o microcontrolador. Assim, a principal preocupação deste manual vai ser na programação por instruções BASIC.

Para mais informações sobre o método de programação por fluxogramas, veja o apêncice sobre fluxogramas.

```
main:
high 0
wait 1
low 0
wait 1
goto main
```

Tutorial 1 – Compreendendo e usando o sistema PICAXE

O *chip* PICAXE, o 'cérebro' do sistema PICAXE, quando comprado sem programa de controlo, não faz absolutamente nada!
O utilizador tem que escrever o programa de controlo no computador e transferi-lo para o microcontrolador PICAXE.
Assim, um sistema PICAXE é constituído por três componentes principais:

O software 'Programming Editor'

Este software é executado num computador e permite-lhe utilizar o teclado deste para escrever programas numa linguagem BASIC simplificada.
Os programas podem ainda ser gerados através do desenho de fluxogramas.

O cabo série

Este é o cabo que liga o computador ao sistema PICAXE. O cabo apenas precisa de ser ligado durante a transferência do programa. Não precisa de estar ligado durante a execução do programa pois o programa fica permanentemente guardado no microcontrolador – mesmo quando se desliga a alimentação!

O chip PICAXE numa placa

O microcontrolador PICAXE executa um programa que tenha sido para ele transferido. Contudo, o integrado precisa de estar montado numa placa que lhe forneça alimentação e possua os outros componentes, a saber, as resistências para comunicação série, a resistência de *reset* e o ressoador.
Esta placa pode ser adquirida ou construída pelo utilizador em *stripboard* ou placa de circuito impresso.

Resumo – procedimentos de programação

1. Escreva o programa no computador utilizando o software Programming Editor.
2. Ligue o cabo de transferência entre o computador e a placa PICAXE.
3. Ligue a fonte de alimentação à placa do PICAXE.
4. Use o *software* Programming Editor para fazer *download* do programa. O cabo pode ser retirado pós a transferência.

O programa começa imediatamente a ser executado no PICAXE. No entanto, o programa pode ser inicializado em qualquer altura premindo o botão de *reset* (caso exista, ou desligando e ligando a alimentação).

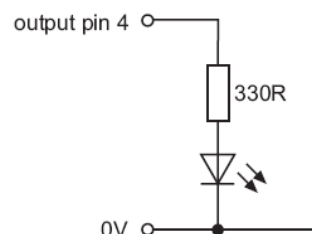
Transferindo o programa BASIC

O programa seguinte liga e desliga a saída 4 em cada segundo. Ao fazer o *download* deste programa o LED deverá acender e apagar de segundo sem segundo.

```
inicio:
high 4
pause 1000
low 4 +
pause 1000
goto inicio
```

Este programa utiliza as instruções **high** e **low** para controlar o pino de saída 4, e usa ainda a instrução **pause** para produzir um atraso (tempo de espera de 1000 ms = 1 segundo).
A última instrução **goto inicio** faz com que o programa 'salte' para a etiqueta (label) **inicio:** no início do programa. Isto significa que o programa está num ciclo infinito. As etiquetas são uasadas nos programas para referenciarem uma dada posição ou localização. Repare que na primeira vez que uma etiqueta (label) é utilizada deve ser seguida do sinal (:). Isso indica ao compilador que a palavra indicada é uma nova etiqueta.

Instruções detalhadas:

1. Ligue o cabo PICAXE ao porto série do computador. Anote em que porto está ligado (normalmente designado COM1 ou COM2).
2. Execute o software Programming Editor.
3. Seleccione o menu View>Options para poder escolher as Opções (deverá aparecer automaticamente).
4. Prima no separador 'Mode' e seleccione o *chip* PICAXE apropriado.
5. Prima no separador 'Serial Port' e seleccione o porto série onde está ligado o

cabo. Prima ‘OK’.
6. Escreva o seguinte programa:

```
inicio:
high 4
pause 1000
low 4
pause 1000
goto inicio
```

(NB repare no sinal (:) a seguir à label ‘inicio’ e os espaços entre as instruções e os números.)
7. Ligue um LED (com uma resistência 330R em série) entre o pino 4 e a massa (0V).
(Assegure-se de que o LED é ligado com a polaridade correcta!).
8. Verifique se o circuito com o PICAXE está ligado ao cabo série, e que a alimentação está ligada.
9. Seleccione PICAXE>Run. Uma barra de *download* deve aparecer à medida que o programa é transferido.
Assim que tiver terminado a transferência o programa entra em execução imediatamente – o LED começa a piscar segundo a segundo.

Tutorial 2 - Usando Símbolos, Comentários & Espaços

Por vezes pode ser difícil lembrar quais os pinos que estão ligados aos dispositivos. A instrução ‘symbol’ pode ser usada no início do programa para renomear as entradas e saídas. Veja-se o exemplo:

```
symbol LED = 4           ‘ renomeia output4 ‘LED’
symbol buzzer = 2        ‘ renomeia output2 ‘buzzer’
principal:               ‘ cria uma label ‘principal’
high LED                 ‘ liga o LED (on)
low buzzer               ‘ desliga o buzzer (off)
pause 1000               ‘ espera 1 segundo (1000 ms)
low LED                  ‘ desliga o LED (off)
high buzzer              ‘ liga o buzzer (on)
wait 1                   ‘ espera 1 segundo
goto principal           ‘ salta para o início
```

Lembre-se que os **comentários** (uma explicação inserida após o sinal (‘)) podem tornar cada linha do programa mais fácil de entender. Os comentários são ignorados pelo computador quando é feita a transferência do programa para o PICAXE.

Uma label (por ex. **principal:** no programa acima) pode ser qualquer palavra (fora as palavras reservadas como ‘switch’ ou ‘pause’, por exemplo), mas devem começar sempre por uma letra. Quando a label é apresentada pela primeira vez ser seguida pelo sinal (:). Este sinal indica ao computador que se trata de uma nova label.
Este programa usa a instrução **wait**. As instruções **wait** e **pause** criam ambas atrasos.
Contudo a instrução **wait** só pode ser usada para segundos, enquanto a instrução **pause** pode ser usada para atrasos mais pequenos (medida em milisegundos (1/1000 do segundo)).

Wait pode ser seguida por um número entre 1 e 65.
Pause pode ser seguida por um número entre 1 e 65535.

É também uma boa técnica de programação o uso de tabulações no início das linhas sem labels, de modo a que as instruções fiquem alinhadas. O termo **‘whitespace’** (espaço em branco) é usado pelos programadores para se referirem a tabulações, espaços e linhas em branco nos programas, que podem tornar a leitura mais fácil.

Nota:
Algumas linguagens BASIC mais antigas utilizavam **‘números de linha** em vez de **labels** nas instruções ‘goto’.
Este sistema é inconveniente pois alterações posteriores no programa, obrigam a alterar a numeração. O sistema de labels (etiquetas) é usado na maioria das linguagens modernas.

Tutorial 3 – Ciclos For…Next

É por vezes muito útil, a repetição de parte do programa um certo número de vezes, como, por exemplo, no piscar de LEDs. Nestes casos pode usar-se um ciclo **for…next**.

O programa seguinte pisca um LED ligado ao pino de saída 1, 15 vezes. O número de vezes que será repetido o código é guardado numa variável de uso geral na memória RAM do PICAXE, utilizando a variável b1 (o PICAXE possui 14 variáveis para usos gerais designadas b0 a b13). Estas variáveis podem ser renomeadas usando a instrução **symbol** para nos lembrarmos mais facilmente.

```
symbol contador = b1            ' define a variável b1 como "contador"
symbol LED = 4                  ' define o pin 4 com o nome "LED"
principal:
      for contador = 1 to 15    ' inicia um ciclo for...next
            high LED            ' liga o pino 7 alto
            pause 500           ' espera 0.5 segundos
            low LED             ' liga o pino 7 baixo
            pause 500           ' espera 0.5 segundos
      next counter              ' fim do ciclo for...next
                                ' repete 15 vezes
end                             ' fim do programa
```

Note ainda como os espaços em branco foram usados para tornar mais clara a sequência de instruções contidas entre as instruções **for** e **next**.

Tutorial 4 – Produzindo sons

Os *buzzers* (bezouros) produzem uma frequência fixa quando ligados.
Contudo, o sistema PICAXE pode criar som de frequências diferentes através da instrução *sound* com um altifalante piezoeléctrico. Todos os chips PICAXE suportam a instrução *sound*, que é destinada a produzir "bips" de aviso.
Recomenda-se o seu uso em vez de bezouros, que consomem mais corrente.

Exemplo de programa para som:

```
inicio:
      sound 2,(50,100)    ' freq 50, duração 100
      sound 2,(100,100)   ' freq 100, duração 100
      sound 2,(120,100)   ' freq 120, duração 100
      pause 1000          ' espera 1 segundo
goto inicio               ' salta para o início
```

Para testar este programa deve colocar um altifalante piezoeléctrico entre o pino de saída (neste caso a saída 2) e 0V.
O primeiro número indica o número do pino (neste caso a saída 2). O número seguinte (entre parêntesis) é o tom, seguido da duração. Quanto mais alto for o tom, mais alto será o som (os valores válidos vão de 0 a 127).

O programa seguinte usa um ciclo *for...next* para produzir 120 sons diferentes.

```
main:
      for b0 = 1 to 120         ' início de um ciclo for...next
            sound 2,(b0,50)     ' produz um som de freq b0
      next b0                   ' ciclo seguinte
end
```

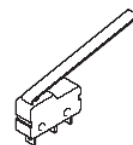
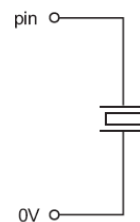
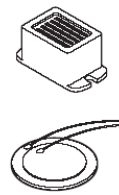
O número armazenado na variável b0 aumenta de 1 em cada ciclo (1-2-3 etc.)
Portanto, ao usar a variável b0 como tom, a nota vai ser modificada em cada ciclo.
O programa seguinte produz a mesma tarefa mas, agora de trás para a frente, usand o como valor de step -1 (em vez do valor +1 por omissão do exemplo acima).

```
main:
      for b0 = 120 to 1 step -1 ' contagem decrescente
            sound 2,(b0,50)     ' produz um som de freq b0
      next b0                   ' ciclo seguinte
end
```

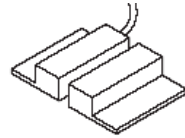
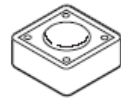
Tutorial 5 – Usando Entradas Digitais

O sensor digital mais simples é um interruptor (microswitch) que possua duas posições, ligado (on) e desligado (off):

Exemplos comuns de sensores digitais são:

- microinterruptores
- interruptores de pressão

• interruptores *reed*
O programa abaixo mostra como usar botões de pressão. Neste programa a saída 4 pisca cada vez que a entrada 3 é premida.

```
inicio:                     ' cria a label 'inicio'
if pin3 = 1 then flash      ' salta se a entrada estiver ligada
goto inicio                 ' caso contrário volta para 'inicio'

flash:                      ' cria a label'flash'
      high 4                ' liga a saída 4 (on)
      pause 2000            ' wait 2 seconds
      low 4                 ' desliga a saída 4 (off)
goto inicio                 ' salto para o início
```

No programa as três primeiras linhas ficam em ciclo permanente. Se a entrada estiver desligada (=0) o programa permanece aqui. Se a entrada ficar alta (=1) o programa salta para a label de nome **'flash'**. Nesse caso a saída 4 liga durante dois segundos e depois desliga voltando ao início.

Repare a sintaxe da linha **if…then** – **pin3** é uma palavra (sem espaços). Isso resulta de pin3 ser o nome de uma variável que contem o dado do pino de entrada respectivo. Nota ainda que após **then** apenas se coloca a label (localização do ponto para onde o programa salta – não são permitidas outras variantes.

Podemos combinar os dados de dois ou mais interruptoes através dos operadores lógicos AND ou OR.

Uma porta AND de duas entradas é programada como

```
if pin2 = 1 and pin3 = 1 then flash
```

Uma porta OR de três entradas é programada como

```
if pin1 = 1 or pin2 = 1 or pin3 = 1 then flash
```

Para ler todo o porto de entrada pode escrever-se

```
let b1 = pins
```

ou

```
b1=pins
```

Para isolar pinos individualmente (por ex. 6 e 7) no porto, temos que mascarar a variável com o operador lógico AND

```
let b1 = pins & %11000000
```

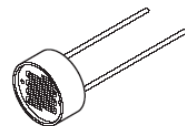
Tutorial 6 – Using Analogue Sensors

Um sensor analógico mede um sinal contínuo como a luz, a temperatura ou a posição. O sensor analógico fornece uma tensão variável contínua. Esta tensão do sinal pode ser representada por um número no intervalo de 0 a 255 (por ex. escuro=0, luz=255).

Exemplos comuns de sensores analógicos são:

• LDR (Light Dependant Resistor)
• Termistores (resistências variáveis com a temperatura)
• Resistências variáveis (potenciómetros)

Light Dependent Resistor (LDR)

A LDR é bem exemplo de um sensor analógico. Pode ligar-se a uma entrada ADC do PICAXE28X (pinos ADC0 a ADC3). Note que apenas estas entradas possuem a capacidade de conversão analógico-digital.

O valor de entrada analógica pode ser guardada numa variável atravé das instrução *readadc10*.
O valor da variável (0 a 1024) pode ser então testado.

O programa seguinte liga um LED se o valor lido for superior a 512 e um LED diferente se o valor for inferior a 128. Para valores entre 128 e 512 ambos os LEDs ficam apagados

```
main:                             ‘ define uma label chamada main
      readadc 1,b0                ‘ lê ADC1 para a variável b0
      if b0 > 512 then top        ‘ se b0 > 512 vai para top
      if b0 < 128 then bot        ‘ if b0 < 128 vai para bot
      low 0                       ‘ caso contrário desliga a saída 0
      low 4                       ‘ e a saída 4
goto main                         ‘ salta para o início

top:                              ‘ define label top
      high 0                      ‘ liga a saída 0
      low 4                       ‘ desliga a saída 4
      goto main                   ‘ salta para o início

bot:                              ‘ define label bot
      high 4                      ‘ liga a saída 4
      low 0                       ‘ desliga a saída 0
      goto main                   ‘ salta para o início
```

Tutorial 7 - Utilização da instrução Debug

Quando se usam sensores analógicos é por vezes necessário calcular um valor de referência (‘threshold’)
(isto é, valores como os 128 e 512 do programa tutorial 6).

A instrução *debug* fornece um modo simples de visualizar “em tempo real”, o valor do sensor, pelo que o valor de referência pode ser assim calculado experimentalmente.

```
main:                             ‘ define uma label designada main
      readadc 1,b0                ‘ lê o canal 1 para a variável b0
      debug b0                    ‘ transmite o valor para o ecrã do PC
      pause 500                   ‘ pequeno atraso
goto main                         ‘ salta para o início
```

Assim que este programa é executado, surge no ecrã do PC uma janela ‘debug’ apresentando os valores de todas as variáveis.
À medida que varia a luz incidente sobre o sensor LDR, o valor da variável mostra o valor lido.
A janela *debug* abre automaticamente após a transferência de um programa que contenha a instrução *debug*. A janela pode ainda ser aberta no menu PICAXE>Debug menu.

Tutorial 8 – Uso do Terminal Série com a instrução Sertxd

Todas as versões PICAXE suportam a instrução *debug*. Contudo, as versões M e X também suportam mensagens série de *debug* mais complexas através da instrução *sertxd*, que envia uma *string* definida pelo utilizador em série para o computador (a uma *baud rate* 4800).

As mensagens podem ser visualizadas através da função Serial Terminal (PICAXE>Terminal menu). O Serial Terminal pode ainda ser aberto sempre que se realiza um *download* através do menu View>Options>Options menu.

```
main:                      ‘ make a label called main
      readtemp 1,b0        ‘ read channel 1 into variable b0
      sertxd (“O valor e’ “,#b0,cr,lf)
      pause 500            ‘ short delay
goto main                  ‘ jump back to the start
```

A instrução *sertxd* transmite a *string* “O valor e´” seguida da *string* ASCII contendo o valor actual da variável b1 (o prefixo # da variável indica uma *string* ASCII que representa o valor correcto transmitido). As constantes CR e LF são valores pré-definidos. (13 e 10) que fazem com que o terminal série mostre uma linha nova para cada valor, de modo a actualizar correctamente o ecrã.
Este programa usa a instrução *readtemp* para ler a temperatura de um sensor de temperatura digital DS18B20, ligado

Tutorial 9 - Sistemas de numeração

Um microcontrolador funciona executando uma enorme quantidade de instruções, no mais curto intervalo de tempo, processando sinais digitais. Estes sinais são codificados no sistema binário – o sinal pode ser **high** (1) ou **low** (0).

Por comparação, lembre-se que o sistema de numeração corrente é o *sistema decimal*. Este sistema de numeração usa dez dígitos, de 0 a 9 para representar valores maiores ou menores.
Contudo, quando se trabalha com microcontroladores é mais fácil pensar em código binário. Isto é particularmente verdade quando se tenta controlar múltiplas saídas ao mesmo tempo.

Um dígito binário é designado por **bit** (binary digit). O sistema PICAXE utiliza 8 bits (1 **byte**), com o bit menos significativo ao lado direito e o mais significativo ao lado esquerdo.

Assim o número binário %11001000 significa que os bits 7,6,3 estão altos (*high*/1) e os outros baixos (*low*/0). O sinal % indica ao compilador que se está a trabalhar em binário, em vez de decimal.

Isso significa que todas as 8 saídas podem ser controladas ao mesmo tempo, em vez de utilizar múltiplas instruções *high* ou *low*. O programa seguinte mostra como apresentar num *display* de 7 segmentos uma contagem de 0 to 9.

```
main:
      let pins = %00111111            ‘ dígito 0
      pause 250                       ‘ espera 0.25 segundos
      let pins = %00000110            ‘ dígito 1
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01011011            ‘ dígito 2
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01001111            ‘ dígito 3
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01100110            ‘ dígito 4
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01101101            ‘ dígito 5
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01111101            ‘ dígito 6
      pause 250                       ‘ espera 0.25 segundos
      let pins = %00000111            ‘ dígito 7
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01111111            ‘ dígito 8
      pause 250                       ‘ espera 0.25 segundos
      let pins = %01101111            ‘ dígito 9
      pause 250                       ‘ espera 0.25 segundos
goto main
```

Cada instrução ‘let pins=’ modifica o número de segmentos que é ligado no *display*. Isto é mais rápido, e mais eficiente em termos de memória, do que utilizar muitas instruções high e *low*.

Tutorial 10 - Subrotinas

Uma subrotina é um pequeno programa separado que pode ser chamado a partir do programa principal. Uma vez executada a subrotina, o pograma principal continua a partir da instrução seguinte àquela que chamou a subrotina. As subrotinas são utilizadas para tornar os programas modulares, isto é, dividi-los em pequenas tarefas, fáceis de implementar e compreender.

Por outro lado, as subrotinas de uso geral, depois de afinadas, podem ser reutilizadas noutros programas.

O PICAXE28X aceita até 255 subrotinas.

O programa seguinte utiliza duas subrotinas para implementar as duas secções do programa (‘flash’ e ‘noise’).

```
symbol LED = 4            ‘ renomeia output4 como ‘LED’
symbol buzzer = 2         ‘ renomeia output2 como ‘buzzer’
symbol counter = b1       ‘ define um countador na variável b1
main:                     ‘ cria uma label designada ‘main’
      gosub flash         ‘ chama subrotina flash
      gosub noise         ‘ chama subrotina noise
goto main                 ‘ volta para o início
end                       ‘ fim do programa principal

flash:                    ‘ make a sub-procedure called flash
      for counter = 1 to 25 ‘ start a for…next loop
            high LED      ‘ LED on
            pause 50      ‘ wait 0.05 second
            low LED       ‘ LED off
```

```
            pause 50      ‘ wait 0.05 second
      next counter        ‘ next loop
return                    ‘ retorno da subrotina

noise:
      high buzzer         ‘ liga o bezouro
      pause 2000          ‘ espera 2 segundos
      low buzzer          ‘ desliga o bezouro
return                    ‘ retorno da subrotina
```

Este segundo programa mostra como uma variável pode ser usada para transferir informação para dentro da subrotina. Neste caso, a variável b2 é usada para dizer ao microcontrolador para piscar 5 vezes o LED e depois 15 vezes.

```
symbol LED = 4            ‘ renomeia output4 como ‘LED’
symbol counter = b1       ‘ define um countador na variável b1

main:                     ‘ cria label desinada ‘main’
      let b2 = 5          ‘ carrega b2 com 5
      gosub flash         ‘ chama subrotina flash
      pause 500           ‘ espera
      let b2 = 15         ‘ carrega b2 com 15
      gosub flash         ‘ chama subrotina flash
      pause 500           ‘ espera
goto main                 ‘ volta ao início
end                       ‘ fim do programa principal

flash:                    ‘ cria subrotina flash
      for counter = 1 to b2 ‘ inicio do ciclo for…next
            high LED      ‘ LED on
            pause 250     ‘ espera 0.25 segundos
            low LED       ‘ LED off
            pause 250     ‘ esperat 0.25 segundos
      next counter        ‘ ciclo seguinte
return                    ‘ retorno da subrotina
```

Tutorial 11 - Usando Interrupts

Um *interrupt* é um caso especial de subrotina. A subrotina ocorre imediatamente após verificar-se uma combinação particular dos sinais nas entradas.

Uma *polled interrupt* é um modo rápido de reagir a uma combinação particular de entradas.

Trata-se do único tipo de *interrupt* disponível no PICAXE. O porto de entrada é lido entre a execução de duas instruções, e durante a execução de uma instrução como *pause*.
Se a combinação de entradas prevista se verificar (verdade), dá-se um ‘gosub’ para a subrotina *interrupt*, que é executada imediatamente. Quando terminar a execução da subrotina dá-se o regresso ao programa principal, na exacta localização onde se deu a interrupção.

A condição das entradas que produzem a interrupção é um padrão de ‘0’s e ‘1’s no porto de entrada, mascarados pelo *byte mask*. Portanto, quaisquer bits mascarados com ‘0’ no *byte mask* serão ignorados.
por ex.
para se produzir um *interrupt* quando o input1 está alto (*high*)

```
setint %00000010,%00000010
```

para se produzir um *interrupt* quando o input1 está baixo (*low*)

```
setint %00000000,%00000010
```

para se produzir um *interrupt* quando o input0 está alto (*high*), o input1 alto (*high*) e o input 2 baixo (*low*)

```
setint %00000011,%00000111
```

etc.

Em cada instante apenas pode haver um padrão de entradas activo (condição a verificar).

Para desactivar o *interrupt* execute a instrução SETINT com o valor 0 como *mask byte*.

Notas:

1) Qualquer programa que possua uma instrução SETINT deve ter a correspondente subrotina *interrupt:* (devidamente terminada pela instrução *return*).

2) Quando se verifica o interrupt, as interrupções ficam desactivadas. Portanto, para voltar a activar as interrupções deve usar-se a instrução SETINT no fim da subrotina *interrupt:*. A interrupção não fica activa senão depois da execução da instrução *return*.

3) Se a interrupção for reactivada e a condição de interrupção não for reinicializada na subrotina, verificar-se-á nova interrupção logo após a execução da instrução *return*.

4) Após a execução do código constante da subrotina *interrupt*, o programa continua a executar-se na instrução seguinte do programa principal. No caso de a interrupção se ter verificado quando da execução de instruções como *wai*t ou *pause*, o tempo restante é ignorado e o programa continua na instrução seguinte.

Explicações mais detalhadas sobre a instrução SETINT.

A instrução SETINT deve ser seguida por dois números – um número ‘a comparar com’ (input) e uma ‘máscara de entradas’ (mask).
É habitual apresentar esses números na forma binária, pois torna mais claro quais os pinos em questão. No formato binário a entrada7 (input7) fica à esquerda da entrada0 (input0).
O segundo número, a ‘máscara de entradas’, define quais os pinos que serão testados para verificar se a interrupção se deve dar …

- %00000001 testa a entrada no pino 0
- %00000010 testa a entrada no pino 1
- %01000000 testa a entrada no pino 6
- %10000000 testa a entrada no pino 7
- etc

Podem combinar-se várias condições a verificar em diferentes pinos ao mesmo tempo...

- %00000011 testa as entradas nos pinos 1 e 0
- %10000100 testa as entradas nos pinos 7 e 2

Tendo decidido quais os pinos que vão ser usados para produzir o *interrupt*, o primeiro número (valor das entradas) define se queremos que a interrupção ocorra quando esses pinos fiquem altos (1) ou baixos (0).

Uma vez a instrução SETINT activa, o PICAXE monitoriza os pinos especificados na máscara de entrada ‘input mask’, onde estão presentes ‘1’, ignorando os outros pinos.

Uma máscara de entrada %10000100 verificará os pinos 7 e 2 criando um valor %a0000b00 onde o *bit* ‘a’ será 1 se o pino 7 estiver alto (*high*) e 0 se estiver baixo (*low*), e o *bit* ‘b’ será 1 se o pino 2 estiver alto (*high*) e 0 se estiver baixo (*low*).

O número ‘a comparar com’ (inputs), o primeiro operando da instrução SETINT, é o número com o qual e valor criado vais ser comparado e, se os dois se igualarem, então ocorrerá o interrupt, caso contrário nã haverá interrupt.

Se ‘máscara de entradas’ for %10000100, serão os pinos 7 e 2 testados, pelo que podemos criar valores ‘a comparar com’ do tipo...

- %00000000 Pino 7 = 0 e pino 2 = 0
- %00000100 Pino 7 = 0 e pino 2 = 1
- %10000000 Pino 7 = 1 e pino 2 = 0
- %10000100 Pino 7 = 1 e pino 2 = 1
-

Assim, se pretender gerar um interrupt quando o pino 7 ficar alto e o pino 2 ficar baixo, a máscara a usar será %10000100 e o valor ‘a comparar com’ deverá ser %10000000, donde a sintaxe da instrução SETINT...

- SETINT %10000000,%10000100

O *interrupt* ocorrerá quando, e apenas quando, o pino 7 ficar alto e o pino 2 ficar baixo.

Exemplo:

```
setint %10000000,%10000000
' activa o interrupt quando o pin7 fica alto (high)
loop:
      low 1 ' switch output 1 off
      pause 2000                          ' espera 2 segundos
goto loop                                 ' volta ao início

interrupt:
      high 1                              ' liga a saída 1
      if pin7 = 1 then interrupt          ' fica em ciclo aqui até que a
                                          ' condição desapareça
      pause 2000                          ' espera 2 segundos
      setint %10000000,%10000000          ' reactiva o interrupt
return                                    ' retorno da subrotina
```

Neste exemplo um LED na saída 1 acenderá imediatamente quando a entrada ficar alta.
Com uma instrução *if pin7 =1 then*.... o programa poderia ter que esperar até dois segundos para acender o LED pois a instrução *if* não é executada durante o tempo de espera produzido pela instrução *pause 2000* do programa principal (o programa standard é apresentado a seguir para termo de comparação).

```
loop:
      low 1                         ' desliga a saída 1
      pause 2000                    ' espera 2 segundos
      if pin7 = 1 then sw_on
      goto loop                     ' volta ao início

sw_on:
      high 1                        ' liga a saída 1
      if pin7 = 1 then sw_on
                                    ' fica em ciclo aqui até que a
                                    ' condição desapareça
      pause 2000                    ' espera 2 segundos
      goto loop                     ' volta ao programa principal
```